Learning to Control Reconfigurable Staged Soft Arms

Austin Nicolai[1], Gina Olson[1], Yiğit Mengüç[1,2], Geoffrey A. Hollinger[1]

***Abstract*— In this work, we present a novel approach for modeling, and classifying between, the system load states introduced when constructing staged soft arm configurations. Through a two stage approach: (1) an LSTM calibration routine is used to identify the current load state then (2) a control input generation step combines a generalized quasistatic model with the learned load model. Our experiments show that accounting for system load allows us to more accurately control tapered arm configurations. We analyze the performance of our method using soft robotic actuators and show it is capable of classifying between different arm configurations at a rate greater than 95%. Additionally, our method is capable of reducing the end-effector error of quasistatic model only control to within 1 cm of our controller baseline.**

I. Introduction

Compared to traditional rigid robots, soft robots provide a unique set of challenges and benefits for applications in low load, high flexibility tasks. Soft robots are used across a variety of domains, including grasping [1], [2], prosthetic devices [3], [4], and biologically inspired locomotion [5], [6], [7], [8]. A common morphology seen in soft robots for these applications is a long, thin bending arm. Soft bending arms are most commonly constructed with three or more longitudinal actuators in parallel, where bending is controlled by lengthening or shortening actuators. Biological systems, such as octopuses, use a similar arm architecture, with the arm tapering towards the tip [9]. This tapering allows for a trade-off between strength (near the base) and flexibility (near the tip). Current state-of-the-art dynamic models for soft bending arms require experimental fits to determine their physical parameters. While these are suitable for controlling individual arms, varying the arm configuration requires a new set of parameters to be determined. A more generalized model is desirable to take advantage of the fact that arm configurations comprised of multiple link segments can be reconfigured to suit different scenarios. The quasistatic model proposed in [10] combines conservation of energy with static equilibrium to provide a generalizable formulation; however, this formulation assumes an unloaded arm.

In order to work with the types of staged arm architectures previously mentioned, we need to be able to account for the system load introduced when connecting multiple longitudinal actuators in sequence. In this work, we propose an approach that augments the generalizable model in [10] with a data-driven, learned load model that accounts for the violation of the no-load assumption. With this approach, no additional learning or parameter fitting is required when switching between arm configurations. Another benefit is that the generalizable model provides a lower bound for our input control pressures that would otherwise not be available when learning the entire control model. In our approach, we first determine the staged arm configuration in use via a long short-term memory (LSTM) based, single-shot calibration routine using an OptiTrack system. After calibration, only simple pressure based on-off control is required to control the arm. We validate the proposed approach on multiple staged soft arms showing that it is able to accurately identify the arm configuration in use. Additionally, we show that the learned load model is able to compensate for the system load across various staged arm configurations.

In summary, we present a novel control architecture for reconfigurable staged soft arms that:

- Compensates for the system load introduced when connecting multiple longitudinal soft actuators in sequence.
- Leverages the temporal nature of the arm motion to accurately identify the current arm configuration.
- Requires no additional learning or parameter fitting when switching between arm configurations.

The remainder of this paper is organized as follows. We discuss related kinematic modeling and data-driven control methods in Section II. Next, we describe our proposed approach in Section III. Our experiments and their results are presented in Section IV. Finally, we conclude and propose avenues for future work in Section V.

II. Related Work

In soft robotics, kinematic modeling approaches commonly utilize the constant curvature approximation. An overview of these techniques are presented by Webster and Jones [11]. When actuator characteristics can be more easily measured (e.g., cable driven arms) more precise geometric models can be used [12], [13]. Controllers that incorporate sensory feedback have also been proposed, including both single camera [14] and multiple camera [15] controllers.

In recent years, many data-driven learning based approaches have been proposed for controlling soft robots. An early model-free implementation of a static controller was proposed by Giorelli et al. in which the inverse statics of a cable driven soft robot were directly learned with a neural network [16]. This approach was validated on physical hardware for a 2 degree of freedom (DoF) [17] and 3 DoF [18] soft manipulator and was shown to outperform the more complex model used for comparison. An approach learning

This work was funded in part by the National Science Foundation National Robotics Initiative (NSF NRI award IIS-1734627).

[1]Collaborative Robotics and Intelligent Systems (CoRIS) Institute, Oregon State University, Corvallis, Oregon 97331

[2]Facebook Reality Labs, Redmond, Washington

Authors' email: {nicolaia, olsongi, yigit.menguc, geoff.hollinger}@oregonstate.edu

the inverse kinematics of a soft robot was presented by Thurunthel et al. in [19], [20]. In this approach, the problem was formulated as a differential inverse kinematics problem using local mapping which allows for multiple solutions globally. More recently, Thurunthel et al. have proposed an approach that additionally incorporates end-effector feedback for learning the inverse kinematics [21]. They show that their algorithm can deal with stochasticity and exhibits adaptive behavior in an unstructured environment. Finally, Bruder et al. have proposed using Koopman operator theory to learn linear dynamical models for a non-linear soft robot arm [22]. The benefit to this method is that in the lifted linear state, the optimization problem is convex allowing for use with model predictive control techniques. While these data-driven methods perform well, they do not extend to generalizable arm configurations which is desirable for reconfigurable staged arm architectures.

In addition to solely data-driven methods, hybrid controllers that combine both model-based and model-free approaches have been proposed. Jiang et al. have proposed a hybrid approach that first uses a cost function to transform between task and configuration space [23]. Once a pose has been identified, a neural network is used to map curvature and arc length to control pressures. Another approach proposed by Reinhart et al. observes that errors in the constant curvature model reduce the accuracy of the inverse kinematic controller [24]. To reduce this error, they model the error with a neural network and show that their hybrid method is capable of reducing tracking error for their arm. In our work we use a similar hybrid technique, but rather than learning to compensate for real world model error, we learn a load model that allows us to further generalize the quasistatic model presented in [10] to various staged soft arm configurations.

III. Method

The quasistatic model proposed in [10] is capable of providing a mapping between control input and soft arm states for individual unloaded link segments as given by

$$u = P_{qs}(\kappa, t), \tag{1}$$

where P_{qs} represents the quasistatic model and u, κ, and t represent the input control pressure, soft arm state, and soft arm width, respectively. In our proposed approach, we learn a model for the system load introduced when combining these individual link segments into a staged arm configuration. Following the previous notation, this extends our mapping function to

$$u_t = P_{qs}(\kappa, t) + P_{load_t}(s, \kappa), \tag{2}$$

where t represents the width of a given link segment in the staged arm and u_t, P_{load_t}, and s represent the input control pressure, learned load model, and load state for the link segment of width t, respectively. We note that a load state is defined for each link segment in an arm configuration, and (in this work) does not change over time.

Our approach uses a fully connected deep network to learn the load model and an LSTM to identify the current load state. While all arm configurations in this work are used during training, in general only a subset of all possible arm configurations (equal to the number of unique load states) need to be used. A block diagram overview of our system can be seen in Figure 1.

A. Load State Estimation

Every arm configuration used in this work produces a unique load state for each link segment in the arm. This allows us to characterize the load state for each link segment by simply determining which arm configuration is currently in use. Additionally, this enables us to identify the current configuration by only looking at the base link of the arm.

We classify which configuration the arm is in by looking at how the base link control input, u_{base}, and state, κ_{base}, change with respect to time. In the absence of external disturbances, κ_{base} changes smoothly as u_{base} is increased.

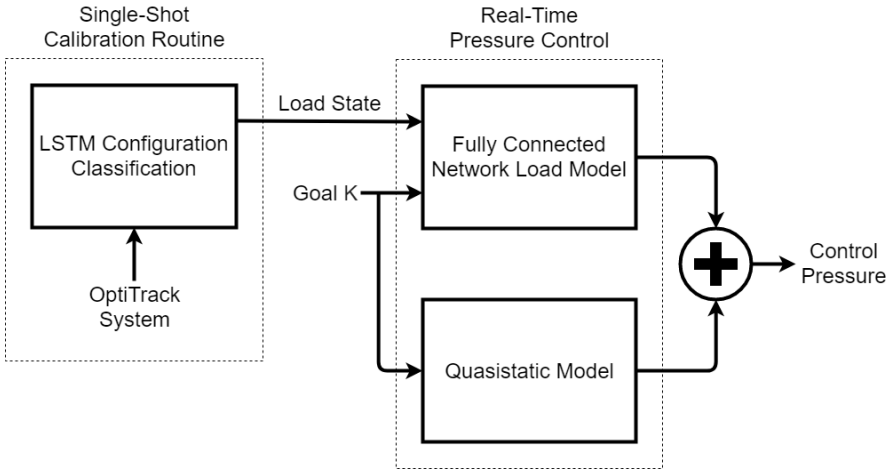


Fig. 1: A block diagram of the architecture proposed in this work. The single-shot calibration routine is only performed once per arm. During operation, real-time pressure control is performed for each individual link segment in the arm. For each segment, an input control pressure is generated using the predicted load state and desired arm curvature state (goal κ). Only the calibration routine requires use of the OptiTrack system.

The result of this is that the pair $(u_{base}, \kappa_{base})_i$ is highly predictive of the pair $(u_{base}, \kappa_{base})_{i+1}$. As such, we use an LSTM based network for classification. To do this, we perform a single-shot calibration routine using an OptiTrack system. This calibration routine is comprised of a single sweep of the arm through its entire valid pressure range. During this sweep, $(u_{base}, \kappa_{base})$ pairs are obtained from the OptiTrack and actuators, respectively. These pairs of data are passed into the LSTM after which a configuration prediction is produced by selecting the class with highest probability.

Since an entire arm sweep is used as classification input, it is time prohibitive to acquire training data solely by hand. Instead, we augment our data by stitching together segments of recorded sweeps to form training sweeps, as detailed in Algorithm 1. In directly stitching these segments together without smoothing, we're able to approximate real world disturbances (e.g., friction, manufacturing imperfections) allowing our classifier to be more robust.

B. Control Input Mapping

We decompose the problem of mapping soft arm states to control inputs into a quasistatic model mapping and load mapping: $u_t = P_{qs}(\kappa, t) + P_{load_t}(s, \kappa)$. This has several benefits. In the planar case, adding a load to the system will increase the control pressure required to achieve a given link curvature state. This means the quasistatic model provides us with a lower bound for the control input. In only learning a portion of the overall state to control input mapping, we minimize the potential impact of incorrect predictions by the network. Since the actuator upper bound pressure ratings are known, we can guarantee a safe control input.

Algorithm 1: Training Data Generation

```
// recorded: recorded data sequences
// N: training sequences to generate
// training: generated data sequences
Input: recorded, N
Output: training
// set of all possible segment lengths
seg_len = {20% 30% 40%}
for i = 1 to N do
    start_idx = 0%      // beginning of sequence
    while start_idx < 100% do
        len <-R- seg_len          // randomly select
        seq <-R- recorded         // randomly select
        end_idx = start_idx + len
        if end_idx ≥ 100% then
            end_idx = 100%  // cut segment short
        end
        // append the segment to the generated
        sequence and then update the index
        segment = seq[start_idx:end_idx]
        training_i.append(segment)
        start_idx = end_idx
    end
end
```

Quasistatic Model: The quasistatic model developed previously [10] assumes constant curvature (Figure 2), an always planar cross section and no loads applied to each individual arm segment. Only one actuator in each segment is pressurized at a time. We calculate curvature by combining moment equilibrium and conservation of energy with a kinematic constraint. The pressurized actuator generates a force F_P that is dependent on the known pressure P and unknown strain ε_P, while the passive actuator force F_E depends only on that actuator's strain ε_E . The moment balance for an arm of width t, summed about the unknown location of the neutral axis, h_P is

$$\Sigma M = -F_P(\varepsilon_P, P)(-h_P) + (-F_E(\varepsilon_E))(t - h_P) = 0. \quad (3)$$

Each actuator has a stored strain energy U_P and U_E, and the pressurized actuator produces work W_P. The energy balance for each segment is

$$\Sigma E = W_P(P, \varepsilon_P) - U_P(\varepsilon_P) - U_E(\varepsilon_E) = 0. \quad (4)$$

The system of equations is closed through a kinematic relationship, which relates the pressurized and passive actuators strains geometrically:

$$\varepsilon_P = -h_P \kappa, \quad (5)$$

$$\varepsilon_E = (t - h_P)\kappa. \quad (6)$$

The actuator force, work and strain energy are calculated from experimental force characterizations [10].

Load Model: In order to learn the mapping for the load model we utilize a fully connected deep network. That is,

$$P_{load_t}(x) = f(Wx + b), \quad (7)$$

where the weight matrix, W, and offset vector, b, are learned parameters; $f(\cdot)$ represents a non-linear activation function; and the input x is given by (s, κ). An individual deep network is trained for each of the link segment widths used in our arm configurations. We note that this scales linearly with the number of link segment widths available. Increasing the length of the overall arm (as measured in number of link segments) does not require additional networks to be trained, but rather increases the number of load states to model.

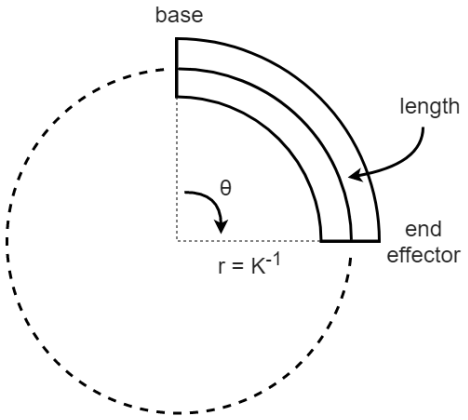


Fig. 2: An illustration of the constant curvature assumption.

Training data is generated by recording the individual link segment actuator pressure states at various curvatures. The quasistatic model value is subtracted from the recorded pressure to obtain the load model value to learn. To efficiently generate training data, we perform arm sweeps that cover the entire valid pressure range of the actuators. This is beneficial because every sweep of the arm performs slightly differently due to an accumulator tube and real world noise (e.g., friction and manufacturing imperfections). This slight performance variation naturally obtains data samples at different curvature values throughout the valid pressure range.

The entire training data set is generated by combining all recorded data points into a single distribution. For each link width t, we have a set of load states, $\{\mathbf{s}\}$, and a set of recorded data pairs, $\{\mathbf{u},\, \boldsymbol{\kappa}\}$. We model this distribution as

$$\mathbf{u}_t = P_{load_t}(\mathbf{s}, \boldsymbol{\kappa}) + N(\mu, \sigma^2), \tag{8}$$

where $\mathbf{u}_t$ are the recorded input pressures for link width t, $P_{load_t}(\mathbf{s}, \boldsymbol{\kappa})$ is the true load model we wish to learn, and $N(\mu, \sigma^2)$ approximates the real world noise observed.

IV. Experiments and Results

We demonstrate the capability of our approach through three experiments. In all three experiments, we use four different link segment widths: 80 mm, 65 mm, 50 mm, and 35 mm. Staging these four widths in tapered configurations (connected via 3D printed spacing plates) provides us with three unique arm configurations: 80-65-50mm, 80-65-35mm, and 80-50-35mm. Each arm segment is made of two actuators joined by six evenly spaced radial support plates. Actuators were made using EcoFlex 00-30, had a wall thickness of 1.5 mm, internal diameter of 6 mm, length of 240 mm, and were reinforced with polyester expandable sheathing (McMaster #9284K2). For full manufacturing details, we refer the reader to [10]. Links were controlled using real-time feedback as detailed in Section IV-A.

In Experiment 1, we compare two classification methods: a fully connected deep network and an LSTM. We use precision and recall as our metrics for success. In general, precision and recall are calculated as

$$Precision = \frac{T_p}{T_p + F_p} \tag{9}$$

$$Recall = \frac{T_p}{T_p + F_n}, \tag{10}$$

respectively, where T_p represents true positives, F_p represents false positives, and F_n represents false negatives.

In Experiment 2, we measure accuracy in curvature space using the mean and maximum error (with standard deviation). For each segment width, all load cases are averaged into the reported metric. We compare several methods:

- **Curvature Control (baseline)** uses the OptiTrack system for state feedback, directly controlling the arm to the desired curvature. This represents the maximum accuracy of our controller.
- **Individual Curve Fit Control** uses individually fit curves (6^{th} order polynomial) to model each load state of each segment width. This approach scales combinatorically with both the number of link widths and arm configurations considered.
- **Single Surface Fit Control** uses a single surface fit (6^{th} order polynomial) to model all load states for every segment width. This approach scales linearly with the number of link widths.
- **Deep Network Control** uses a single trained deep network to model all load states for each segment width. This also scales linearly with the number of link widths.

In Experiment 3, mean and maximum errors of the end-effector are reported in Euclidean space (with standard deviation) to provide a more intuitive understanding of the quality of control. We compare three methods:

- **Curvature Control (baseline)** represents the maximum accuracy of our controller.
- **Quasistatic Model Only Control** uses only the model proposed in [10] to control the arms and does not account for load.
- **Quasistatic Plus Load Model Control** uses the proposed approach to control the arm, accounting for load using a fully connected deep network.

A. Hardware Setup

Full arm control was achieved by controlling individual link segments with on-off feedback control. Our custom control board is capable of both pressure and curvature control via real-time feedback. We note that our proposed approach operates using only pressure feedback. Experiments 2 and 3 compare against direct curvature control as a means of comparing against the maximum accuracy of our controller.

The control board is comprised of a diaphragm pump, solenoid valves, pressure sensors, and an Arduino. The single diaphragm pump provides airflow to the entire system while the solenoid valves control whether or not individual link segment actuators inflate, deflate, or maintain pressure. The Arduino controls the pump speed and solenoid states. MATLAB is used as to interface with both the control board and the OptiTrack system. System feedback is available in two forms: pressure and curvature. Pressure feedback is obtained via onboard Honeywell Basic ABP Series sensors. Curvature feedback is obtained from the positions of retroreflective markers as given by the OptiTrack system.

B. Experiment 1: Configuration Classification Accuracy

We compare our proposed LSTM architecture against a standard, fully connected deep architecture. While both architectures receive the same input data, the fully connected architecture represents a brute force approach that doesn't explicitly leverage the temporal nature of our data. The specific architecture parameters can be seen in Tables I and II. The layer naming convention denotes both the type of layer and number of units (i.e., *lstm30* represents an LSTM layer with 30 units). We train and validate on 600 and 150 sweeps per arm configuration, respectively (total: 1800 and 450). The networks were trained for up to 500 epochs (with early stopping) using the Adam optimizer [25].

For each arm configuration, we perform 25 classification trials. Each trial involves placing the staged arm in the OptiTrack staging area and performing a single sweep of the base link segment's full pressure range (0-12 psi). The recorded $(u_{base}, \kappa_{base})$ pairs are passed through the trained classifiers and the load state prediction for each link in the staged arm, $\hat{s}$, is given by the maximum class probability.

Full results from the experiment can be seen in Table IV. We can see that both network architectures were able to achieve high classification accuracy, with the LSTM slightly outperforming the fully connected deep network. In both cases, the 80-65-50mm configuration was easiest to classify. This is likely because the 35 mm segment is the lightest, and it is easy to detect the configuration without that segment. Classifying between the 80-65-35mm and 80-50-35mm configurations proved more difficult as the difference in motion between them is more slight. An interesting result is that the LSTM network architecture required far fewer trainable parameters (11,373 vs. 250,883) and converged faster (287 vs. 326 epochs) to achieve the reported results, highlighting the benefit of explicitly leveraging the data temporality.

C. Experiment 2: Load Model Accuracy

For this experiment, ground truth load states are given to both the deep network and surface fit to directly measure the quality of the load model learned. The deep network structure can be seen in Table III. We train and validate on 1000 and 200 data points per load case (total: 3000 and 600 for the 80 mm segment, 2000 and 400 for the 65 mm segment, 1000 and 200 for the 50 mm segment). The network was trained for up to 200 epochs using the Adam optimizer [25].

We control each loaded segment (i.e., all segments except the 35 mm) to 25 distinct curvatures equally spaced along the segment's range of motion. We note that this range of motion changes for each segment width and load condition, and thus different curvatures are chosen for each condition.

The results of this experiment can be seen in Table V. Here, we can see that direct curvature control provides a very accurate baseline. The individual curve fits and our proposed method perform accurately (and similarly, except for the 65 mm segment). While the performance is similar, the individual curve fit method scales combinatorically with the number of link widths and arm configurations used. Despite the single surface fit performing the worst in all cases, the gap between it and other methods decreased as the number of load cases modeled decreased (i.e., segment width decreased). This makes sense since for fewer load cases, the modeling problem is more straightforward.

We observed the 65 mm segment to experience inconsistent performance in the low pressure region. Notably, this affected the performance (both mean and maximum error rates) of the individual curve fits more than our proposed method. This is likely because the deep network inherently models functions more complex than a 6^{th} order polynomial.

D. Experiment 3: End-Effector Accuracy

In this experiment, we first perform the single-shot calibration routine using only the LSTM classifier. The load state prediction is then used for the remainder of the experiment, even if misclassified, so as to fully capture the accuracy of the proposed system. The load state and desired curvature state are then used to generate control inputs, $u_t = P_{qs}(\kappa_{goal}, t) + P_{load_t}(\hat{s}, \kappa_{goal})$ for each link in the arm configuration. We control the arm to 25 distinct, randomly generated arm states within the reachable workspace for all arm configurations and measure the resulting end-effector location.

Full results can be seen in Table VI. As expected, the curvature control baseline is the most accurate. Even though the mean error appears to be somewhat high, we note that this error largely comes from the shortening of the base curve of the link segments as they bend. While the control is highly accurate in curvature space (as shown in Experiment 2), as the link bends and shortens, error is introduced along the segment arc. This is further evidenced by the low standard deviation in the results, demonstrating the consistency of both the baseline and our approach. The link shortening can be seen in Fig. 3 and is further discussed in [10].

From the table, we can see that the quasistatic model only control performs poorly. This illustrates that without accounting for the system load, the benefits of a staged arm configuration (e.g., increased flexibility near the tip) cannot be taken advantage of. This mode of control performs poorly enough that it would not be suitable for many real world tasks. As in Experiment 2, we can see that our proposed approach performs closely to the curvature control baseline. For all arm configurations, our proposed method achieves an accuracy within 1 cm of the curvature control baseline.

V. Conclusion

We presented a novel approach capable of learning to model the system load introduced when constructing tapered staged soft arm configurations that is not accounted for in the generalized kinematic model. Our proposed approach is capable of modeling multiple load states allowing the staged arms to be reconfigured with no additional training required. Experiments on soft robotic hardware show that our approach correctly identifies the arm configuration at a rate greater than 95%. Additionally, our method reduces the end-effector error of quasistatic model only control to within 1 cm of our controller baseline. Our approach enables the use of tapered staged arm configurations allowing us to take advantage of their increased flexibility.

These results suggest a couple of avenues for future work. First, the classification step can be extended to predict a segment's general load state by changing the LSTM prediction from classification to regression. This would allow the arm to work with arbitrary external loads (e.g., manipulating an object). Second, to further increase the real world capabilities, we can incorporate soft sensors on the individual link segments. Link segment curvature feedback can be obtained from these, removing the need for the OptiTrack system.

VI. Acknowledgments

We would like to thank Scott Chow and Christopher Bollinger for their help troubleshooting code.

TABLE I: LSTM Classifier Architecture

Layer	Activation	Dropout
lstm30	ReLU	$\sim$
lstm30	ReLU	$\sim$
dense3	Softmax	$\sim$

TABLE II: Fully Connected Classifier Architecture

Layer	Activation	Dropout
dense512	ReLU	20%
dense256	ReLU	20%
dense64	ReLU	20%
dense3	Softmax	$\sim$

TABLE III: Fully Connected Load Model Architecture

Layer	Activation	Dropout
dense128	ReLU	10%
dense32	ReLU	10%
dense1	Linear	$\sim$

TABLE IV: Classification Accuracy Results. Both the mean and individual class metrics are reported. The individual classes are specified by the arm configuration details (mm).

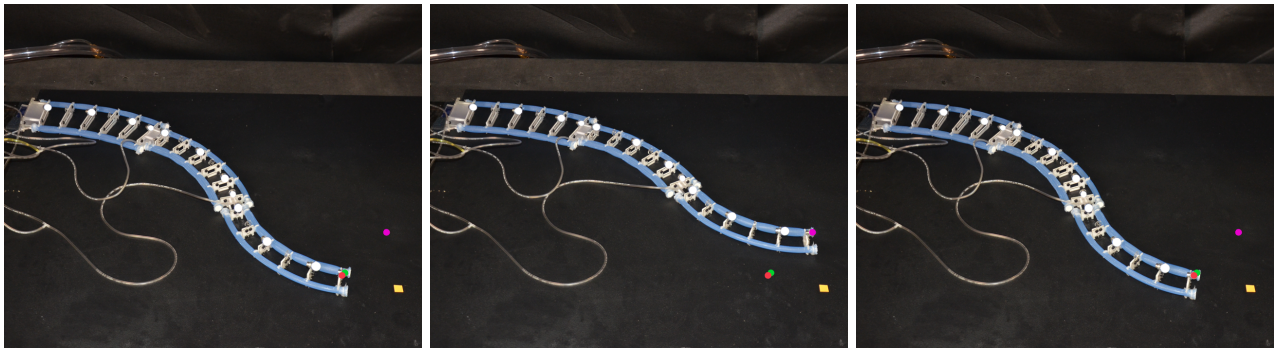
	Precision				**Recall**			
	Mean	**80-65-50**	**80-65-35**	**80-50-35**	**Mean**	**80-65-50**	**80-65-35**	**80-50-35**
Fully Connected	94.85%	100.0%	88.89%	95.65%	94.67%	100.0%	96.0%	88.0%
LSTM (proposed)	97.33%	100.0%	96.0%	96.0%	97.33%	100.0%	96.0%	96.0%

TABLE V: Individual Link Control Accuracy Results. Details regarding each method listed can be found in Section IV.

	80 mm Segment		**65 mm Segment**		**50 mm Segment**	
	Mean Error (m^{-1})	**Max Error (m^{-1})**	**Mean Error (m^{-1})**	**Max Error (m^{-1})**	**Mean Error (m^{-1})**	**Max Error (m^{-1})**
Curvature Control (baseline)	0.015 ± 0.011	0.047	0.022 ± 0.016	0.062	0.026 ± 0.017	0.079
Individual Curve Fit Control	$\mathbf{0.037 \pm 0.021}$	**0.086**	0.189 ± 0.152	0.552	0.076 ± 0.047	0.160
Single Surface Fit Control	0.071 ± 0.041	0.175	0.204 ± 0.177	0.551	0.082 ± 0.052	0.194
Deep Network Control (proposed)	$\mathbf{0.037 \pm 0.028}$	0.123	$\mathbf{0.103 \pm 0.108}$	**0.403**	$\mathbf{0.072 \pm 0.041}$	**0.154**

TABLE VI: Full Arm Control Accuracy Results. Details regarding each method listed can be found in Section IV.

	80-65-50mm Configuration		**80-65-35mm Configuration**		**80-50-35mm Configuration**	
	Mean Error (cm)	**Max Error (cm)**	**Mean Error (cm)**	**Max Error (cm)**	**Mean Error (cm)**	**Max Error (cm)**
Curvature Control (baseline)	2.791 ± 0.380	3.386	3.342 ± 0.402	4.099	3.672 ± 0.626	5.103
Quasistatic Model Only Control	9.215 ± 5.535	17.267	9.572 ± 5.484	18.171	8.110 ± 4.457	15.162
Quasistatic Plus Load Model Control (proposed)	$\mathbf{3.468 \pm 0.512}$	**4.490**	$\mathbf{3.9282 \pm 0.602}$	**5.339**	$\mathbf{4.125 \pm 0.698}$	**5.479**



(a) Curvature control (b) Quasistatic model only control (c) Quasistatic plus load model control

Fig. 3: A comparison of the end-effector error produced by the control methods in Experiment 3. In each image, the theoretical expected end-effector location (before link shortening) is represented by the orange square. The three dots represent the final end-effector location for the different control methods. Curvature control is represented by the green dot, quasistatic model only control is represented by the magenta dot, and quasistatic plus load model control is represented by the red dot.

REFERENCES

[1] H. Zhao, K. O'Brien, S. Li, and R. Shepherd, "Optoelectronically innervated soft prosthetic hand via stretchable optical waveguides," *Science Robotics*, vol. 1, no. 1, 2016.

[2] M. Manti, T. Hassan, G. Passetti, N. D'Elia, C. Laschi, and M. Cianchetti, "A bioinspired soft robotic gripper for adaptable and effective grasping," *Soft Robotics*, vol. 2, no. 3, pp. 107–116, 2015.

[3] G. Singh, C. Xiao, G. Krishnan, and E. Hsiao-Wecksler, "Design and analysis of soft pneumatic sleeve for arm orthosis," *Proc. ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, 2016.

[4] P. Polygerinos, S. Lyne, Z. Wang, L. F. Nicolini, B. Mosadegh, G. M. Whitesides, and C. J. Walsh, "Towards a soft pneumatic glove for hand rehabilitation," *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1512–1517, 2013.

[5] T. Umedachi, V. Vikas, and B. Trimmer, "Softworms: The design and control of non-pneumatic, 3D-printed, deformable robots," *Bioinspiration & Biomimetics*, vol. 11, no. 2, p. 025001, Mar 2016.

[6] C. Branyan, C. Fleming, J. Remaley, A. Kothari, K. Tumer, R. Hatton, and Y. Menguc, "Soft snake robots: Mechanical design and geometric gait implementation," *Proc. IEEE Conference on Robotics and Biomimetics (RoBio 2017)*, pp. 282–289, 2017.

[7] H. Lin, G. G. Leisk, and B. Trimmer, "GoQBot: A caterpillar-inspired soft-bodied rolling robot," *Bioinspiration & Biomimetics*, vol. 6, no. 2, p. 026007, Apr 2011.

[8] M. Calisti, E. Falotico, and C. Laschi, "Hopping on uneven terrains with an underwater one-legged robot," *IEEE Robotics and Automation Letters*, vol. 1, pp. 461–468, 2016.

[9] W. Kier and K. Smith, "Tongues, tentacles and trunks: The biomechanics of movement in muscular-hydrostats," *Zoological Journal of the Linnean Society*, vol. 83, no. 4, pp. 307–324, 1985.

[10] G. Olson, S. Chow, A. Nicolai, C. Branyan, G. Hollinger, and Y. Mengüç, "A generalizable equilibrium model for bending soft arms with longitudinal actuators," *The International Journal of Robotics Research*, DOI:10.1177/0278364919880259, Oct. 2019.

[11] R. J. Webster and B. A. Jones, "Design and kinematic modeling of constant curvature continuum robots: A review," *The International Journal of Robotics Research*, vol. 29, no. 13, pp. 1661–1683, 2010.

[12] F. Renda, M. Cianchetti, M. Giorelli, A. Arienti, and C. Laschi, "A 3D steady-state model of a tendon-driven continuum soft manipulator inspired by the octopus arm," *Bioinspiration & Biomimetics*, vol. 7, no. 2, p. 025006, May 2012.

[13] F. Renda, M. Giorelli, M. Calisti, M. Cianchetti, and C. Laschi, "Dynamic model of a multibending soft robot arm driven by cables," *IEEE Transactions on Robotics*, vol. 30, no. 5, pp. 1109–1122, Oct 2014.

[14] J. M. Croom, D. C. Rucker, J. M. Romano, and R. J. Webster, "Visual sensing of continuum robot shape using self-organizing maps," *Proc. IEEE International Conference on Robotics and Automation*, pp. 4591–4596, 2010.

[15] D. B. Camarillo, C. R. Carlson, and J. K. Salisbury, "Configuration tracking for continuum manipulators with coupled tendon drive," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 798–808, 2009.

[16] M. Giorelli, F. Renda, G. Ferri, and C. Laschi, "A feed-forward neural network learning the inverse kinetics of a soft cable-driven manipulator moving in three-dimensional space," *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5033–5039, 2013.

[17] M. Giorelli, F. Renda, M. Calisti, A. Arienti, G. Ferri, and C. Laschi, "Neural network and jacobian method for solving the inverse statics of a cable-driven soft arm with nonconstant curvature," *IEEE Transactions on Robotics*, vol. 31, no. 4, pp. 823–834, Aug 2015.

[18] ——, "Learning the inverse kinetics of an octopus-like manipulator in three-dimensional space," *Bioinspiration & Biomimetics*, vol. 10, no. 3, p. 035006, May 2015.

[19] T. Thuruthel, E. Falotico, M. Cianchetti, and C. C. Laschi, "Learning global inverse kinematics solutions for a continuum robot," in *ROMANSY 21 - Robot Design, Dynamics and Control*, V. Parenti-Castelli and W. Schiehlen, Eds. Cham, Switzerland: Springer International Publishing, 2016, pp. 47–54.

[20] T. Thuruthel, E. Falotico, M. Cianchetti, F. Renda, and C. Laschi, "Learning global inverse statics solution for a redundant soft robot," *Proc. 13th International Conference on Informatics in Control, Automation and Robotics*, pp. 303–310, 2016.

[21] T. Thuruthel, E. Falotico, M. Manti, A. Pratesi, M. Cianchetti, and C. Laschi, "Learning closed loop kinematic controllers for continuum manipulators in unstructured environments," *Soft Robotics*, vol. 4, no. 3, pp. 285–296, 2017.

[22] D. Bruder, B. Gillespie, C. D. Remy, and R. Vasudevan, "Modeling and control of soft robots using the Koopman operator and model predictive control," *Proc. Robotics: Science and Systems Conference*, 2019.

[23] H. Jiang, Z. Wang, X. Liu, X. Chen, Y. Jin, X. You, and X. Chen, "A two-level approach for solving the inverse kinematics of an extensible soft arm considering viscoelastic behavior," *Proc. IEEE International Conference on Robotics and Automation*, pp. 6127–6133, 2017.

[24] R. Reinhart, Z. Shareef, and J. Steil, "Hybrid analytical and data-driven modeling for feed-forward robot control," *Sensors*, vol. 17, no. 2, 2017.

[25] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.